-------------------- CEN 4072/6070 Software Testing & Verification ------------------

Exam 1 -- Spring 2015 -- Solution Notes

1. a. Combining or overlay of waterfall-style software development on the Scrum (agile development management) framework. Normally used as a pejorative description, as it interferes with (i.e., slows down) the rapid release/delivery of software.
   b. The lines added, modified, or deleted from a file from one version to another. A measure of the changes made to a software component over a period of time. Ken argued that the risks associated with code churn per release are reduced when code deployment is continuous.

2. a. Mocking is creating mock units that simulate the behavior of real units. (It is sometimes used synonymously with "stubbing".)
   b. A unit under test may have dependencies on other (complex) units. Mocking allows you to isolate the behavior of the unit you want to test by replacing the other units with "mocks" that simulate the behavior of the real units.

3. a. Testing undertaken to demonstrate that a system performs correctly is sometimes referred to as validation testing. Testing undertaken to expose defects is sometimes referred to as defect testing.
   b. Validation testing. Beizer suggests that tests are designed and then executed to *demonstrate* the correspondence between an element and its specification.

4. a. re-testing a program using test cases that previously ran without detecting an error to detect problems caused by the adverse effects of program change
   b. actual end-user testing performed within the software development environment
   c. testing conducted without human intervention – e.g., after normal working hours

5. An operational profile provides info regarding the way a system is expected to be used in some environment. This info is needed in order to make inferences about the system's expected reliability in that environment based on statistical testing.

6. d

7. An implicit part of the specification for pow() included the mathematical definition of exponentiation. By seeking out the definition in a standard math dictionary, it was discovered that x**y really *MEANS* different functions depending on what region (point, line, etc.) in the x,y plane the input reflects. The usefulness of this info was that these functional differences could then be modeled as separate non-error Effects in the test case design model.

8. If a domain exception is detected, an integer is stored in errno (by matherr) describing the type of exception. An application wishing to use this info should check for this should set errno to 0 before calling fmod(). If errno is non-zero on return, an error has occurred.

9. The specified functions were: ***fmod,*** which returns the floating-point remainder of the division of x by y, and ***sort,*** which sorts, merges, or sequence checks text files.

   fmod is a relatively simple function that can easily be modeled by a small number (the solution notes identified 5) of mutually exclusive effects combining the specified behaviors of *fmod* and *matherr*. A major advantage of this approach is that it easily allows for the coverage of all combinations of such behaviors by associating test cases with each feasible combination of connected Causes in the Cause-Effect graph. This could be accomplished using a relatively small number of cases for *fmod.*

   In contrast, ***sort*** has a relatively complex interface with a large number of mostly independent behaviors. Attempting to identify mutually exclusive Effects for this function would require identifying an extremely large number of behavioral combinations, where many of the behaviors are independent of one another. Thus, attempting to identify mutually exclusive Effects for sort would probably not be appropriate.

10. a. 12, 3
    b. 6
    c. 2

11. Consider test case #1 with inputs A: true and B: false, and test case #2 with inputs A: false and B: true. These two cases result in condition coverage, but not path coverage. <u>Therefore, condition coverage does not subsume path coverage</u>.

    Now consider test case #3 with inputs A: false and B: false. Test cases #1 and #3 together result in path coverage, but not condition coverage. <u>Therefore, path coverage does not subsume condition coverage</u>.

    Therefore, condition coverage and path coverage are independent.

12. 7X3X2=42
    3X3X2=18

13. a. 4, 4, 4, 3
    b. 9, 5, 3, 2
    c. 8, 4, 2, 1
    d. infinite, 3, 1, 1

14.

	TEST CASES									
CAUSES	1	2	3	4	5	6	7	8	9	10
(1)	F	F	T	T	T					
(2)	F	T	F	F	T					
(3)	T	F	F	T	F					
(4)	F	F	F	T	T					
EFFECT										
(33)	T	T	T	T	T					

15. a. Detecting errors in the same phase as they are introduced – i.e., preventing the migration of errors from any development phase to any subsequent phases.
    b. Proven testing techniques that can be applied early in the development process. The specific examples mentioned are "reviews on critical documents like requirements" and (more generally) "an effective inspections program".

16. a. (1,7), (1,<2,7>), (1,<2,3>), (1,<6,7>), (4,4), (4,<6,3>), (4,<6,7>)
    b. <1,2,3,4>, <1,2,3,5,6,3,4>
    c. <1,2,3,5,6,3>
    d. <4,6,7>
    e. $Y>0 \land X1>0 \land Y-X1>0 \land X1\leq 0 = \text{FALSE}$
    f. $Y>0 \land X1\leq 0 \land Y>0 \land X2>0 \land Y-X2\leq 0$

17. In many organizations, testing is still formally subordinate to (i.e., reports to) development (even though testing is meant to represent the customer and as such primarily owes its allegiance to the customer rather than to development).

18. a. "Keep meetings to one, distinct topic."
    b. Diagram1 depicts a single 2-hour meeting covering 2 distinct topics, each taking one hour to cover. Only tree attendees (stick figures in the intersection of the sets in diagram 1) are interested in both of the 2 topics; all the other attendees are "bored" for one hour. Diagram 2 depicts two separate 1-hour meetings, each focusing on just 1 of the two topics. None of the (stick figure) participants are "bored".

19. In the Fagan model of inspections, individual participants concentrate on trying to understand the element being reviewed, including its intent and logic. The primary goal is NOT to identify errors in this step, although flagrant errors are sometimes found. (It is normally expected that the number of errors found will not be nearly as high as in the following group inspection step.) Participants are also expected to study the ranked distributions of error types found in recent inspections to help prepare them for the inspection meeting.

In contrast, the preparation step in the HP model of inspections focuses on individuals finding items (errors, omissions, inconsistencies, and areas of confusion) and issues (items beyond the scope of the inspection team) in the element being reviewed. This is intended to prepare the participants for the subsequent "logging meeting" in which the primary goal is to log the items and issues discovered, identify false positives, etc.

20. a. An individual preparation stage for the judgment task and a stage during which the group meets in connection with the judgment task
    b. The example is the "Lost in the Desert" problem in which subjects are required to imagine themselves stranded in the desert with a limited number of implements available to them (knife, string, mirror, etc.) which they must rank in order of their survival value. Individuals make their own judgments and then meet to decide a group ranking.

21. T, K, C, L, X, F, W, N, R

Histogram of Raw Scores

```
                                                             *               *           **
                                 $                      *$   **  **       *$ *  *        **         *                         *
                          $      $  *   *$   * * $*$ *$**$   $$* $$ * *   *$*$* $**  $*  $$ *     * $    $  **** $ *  *   *   $ *
1...22222222223333333333444444444455555555556666666666777777777788888888889999999999111111111111111111111111111111111111111111111111111111111111111111
    01234567890123456789012345678901234567890123456789012345678901234567890123456789000000000011111111112222222222333333333344444444445555555555666666
                                                                                    012345678901234567890123456789012345678901234567890123456789012345
                                                                      ^         ^
                                                                      |         |
                                                                    AVG($)    AVG(*)
```

* - CEN 6070 students
$ - CEN 4072 students